

Method and apparatus for determining a set of large
sequences from an electronic data base



The invention relates to the art of sequential analysis in electronic data bases.

5 Background of the invention

In sequential analyses sequences of items are extracted from transactional electronic data. As more and more transactional data are acquired electronically sequential analysis becomes more important.

10 A practical example of a sequential analysis is the clickstream analysis for calculating typical user paths of Web site users. Further examples of sequential analyses are text analysis for extracting characteristic series of words in electronically memorized documents and the analysis of baskets of merchandise taking into account the order of products in trade to define typical product chain purchases. Moreover, sequential analyses are widely used in chemistry and genetics.

20 A great number of specialized methods of sequential analysis are known to date, especially in genetics. However, very few of them are universally applicable. In the simplest case, all variants of possible sequences are studied as regards their frequency. But for reasons of computer capacity that can be done only for small amounts of data. Sequential analysis algorithms based on search trees present an alternative, such as the algorithm "Capri" by Messrs. Lumio (<http://www.spss.com/PDFs/CP2SPCZ-1201.pdf>). Yet their speed, too, is insufficient for analyzing large transaction data. Methods based on the classical analysis of baskets of merchandise offer a quicker alternative and have already been used for sequential analysis of baskets of merchandise (Agrawal R., Srikant R., Mining Sequential Patterns. IBM Almaden Research Center, 650 Harry Road, San Jose).

The invention

It is an object of the invention to indicate a method of and an apparatus for determining a set of large sequences from an electronic data base by which known shortcomings are overcome and, especially,
5 an efficient sequential analysis can be performed on large amounts of data in diverse applications.

According to one aspect of the invention, a method is provided of determining a set of large sequences from an electronic data base comprising a set $D = \{d_1, \dots, d_n\}$ of transactions d_i ($1 \leq i \leq n$) in a
10 computer system with an implemented query module, each of the large sequences on the set D of transactions d_i having a support value greater than or equal to a given support value S , each of the transactions d_i of the set D being a sequence of items of a record $E = \{e_1, \dots, e_m\}$ of items e_j ($1 \leq j \leq m$) and the method comprising the following steps:
15
a) determining a set L_1 of large sequences from the set D of transactions, the large sequences of set L_1 each comprising exactly one item of the record E , and an associated support value S_{L_1} on the sequence D of transactions each being greater than or
20 equal to the given support value S ;
b) determining a set L_2 of large sequences from the set D of transactions, the large sequences of set L_2 each comprising exactly two items of the record E in a respective order R_{L_2} , and an associated support value S_{L_2} on the set D of transactions
25 each being greater than or equal to the given support value S , and nothing but sequences comprising one of the large sequences of set L_1 , as a partial sequence, being taken into account in determining set L_2 ;
c) determining a set L_k ($k > 2$) of large sequences from the set D of
30 transactions, the large sequences of set L_k each comprising exactly k items of record E in a respective order R_{L_k} , and an associated support value S_{L_k} on the sequence D of transactions each being greater than or equal to the given support value S , and nothing but sequences comprising two of the large sequences
35 of set L_{k-1} , as partly overlapping partial sequences, with the

respective order R_{Lk-1} , being taken into account in determining set L_k ; and

- d) repeating step c) for $k = k+1$ and terminating the repetition of step c) when a given termination condition is fulfilled.

5 According to another aspect of the invention an integrated sequential analysis system is provided, comprising:

- an electronic data base comprising a set $D = \{d_1, \dots, d_n\}$ of transactions d_i ($1 \leq i \leq n$), each of the large sequences on the set D of transactions d_i having a support value greater than or equal to a given support value S , each of the transactions d_i of the set D being a sequence of items of a record $E = \{e_1, \dots, e_m\}$ of items e_j ($1 \leq j \leq m$);
- a query module comprising a query means coupled to the data base and a processing means for detecting query parameters and generating queries to the query means;
- 15 - means for determining a set L_1 of large sequences from the set D of transactions, the large sequences of set L_1 each comprising exactly one item of the record E , and an associated support value S_{L_1} on the sequence D of transactions each being greater than or equal to the given support value S ;
- 20 - means for determining a set L_2 of large sequences from the set D of transactions, the large sequences of set L_2 each comprising exactly two items of the record E in a respective order R_{L_2} , and an associated support value S_{L_2} on the set D of transactions each being greater than or equal to the given support value S ,
- 25 and nothing but sequences comprising one of the large sequences of set L_1 , as a partial sequence, being taken into account in determining set L_2 ;
- means for determining a set L_k ($k > 2$) of large sequences from the set D of transactions, the large sequences of set L_k each comprising exactly k items of record E in a respective order R_{L_k} , and an associated support value S_{L_k} on the sequence D of transactions each being greater than or equal to the given support value S , and nothing but sequences comprising two of the large sequences of set L_{k-1} , as partly overlapping partial sequences,
- 30 with the respective order $R_{L_{k-1}}$ being taken into account in determining set L_k ; and
- 35

- means for repeating step c) for $k = k+1$ and terminating the repetition of step c) when a given termination condition is fulfilled.

5 In contrast to the classical basket of merchandise analysis, the invention takes into account the order of items in the transactions. The invention permits automatic evaluations of extensive amounts of electronic data according to characteristic sequences to be made efficiently in terms of time and computer capacity. Sequential analysis of large amounts of data, such as contained in logfiles of heavily frequented Web servers thus becomes possible for the first time.

Drawing

The invention will be described further, by way of example, with reference to the accompanying drawings, in which:

- 15 Fig. 1 is a diagrammatic presentation to explain a sequential analysis method according to the invention in connection with a clickstream analysis;
- Fig. 2 is a diagrammatic presentation of an arrangement for carrying out a sequential analysis method according to the invention;
- 20 Fig. 3 is a flowchart of an embodiment of a sequential analysis method according to the invention in accordance with a first variant; and
- Fig. 4 is a flowchart of an embodiment of a sequential analysis method according to the invention in accordance with a second variant.
- 25

Examples of embodiments

A method of and an apparatus for determining a set of large sequences from an electronic data base will be described below with reference to figs. 1 to 4.

30 First of all, the problem to be solved by sequential analysis will be explained in detail. Let it be assumed that $E = \{e_1, e_2, \dots, e_m\}$,

a record of literal constants, referred to as items. A sequence $\langle s_0, s_1, \dots, s_k \rangle$ be an ordered list of items. $D = \{d_1, d_2, \dots, d_n\}$, a record of transactions, each transaction d_i ($1 \leq i \leq n$) a sequence of items from E so that: $d_i \subseteq E$. An unambiguous identifier, designated "di-ID", is associated with each transaction.

The sequence $\langle a_0, a_1, \dots, a_k \rangle$ is included (partial sequence) in another sequence $\langle b_0, b_1, \dots, b_l \rangle$ when natural non-negative numbers $i_1 < i_2 < \dots < i_k$ exist so that:

$$a_1 = b_{i_1}; a_2 = b_{i_2}; \dots; a_k = b_{i_k}.$$

- 10 It is defined that a sequence A possesses a support $s\%$ on the set of transactions D if it is contained in $s\%$ of all transactions of set D . A sequence is said to be large if its support is not smaller than a given minimum value. A sequence having a length k is called k -sequence. The set of all k -sequences will be referred to below as
- 15 L_k . The set of all k -sequences which potentially may be large is called C_k .

Set E can be mapped unambiguously on a subset of natural numbers, i.e. an unambiguous natural identifier may be assigned to each item from E .

- 20 Using the designations introduced above, the object to be met may be defined as follows: Based on the given sets E and D , all sequences are to be found whose support value is greater than or equal to a given minimum value, i.e. the following set is looked for:

$$L = \bigcup_k L_k$$

- 25 A known solution trial to resolve the above problem resides in listing all the possible sequences of items of set E and counting the transactions containing them. The number of all possible sequences of set E having a length of from 1 to k is found. The power of set E is m . The number of sequences of length 1 equals the number of items
- 30 of set E , in other words m . The number of all possible sequences of

length i of m items is m^i . Then the number of all sequences N equals the sum of the geometric progression to the basis m :

$$N = \sum_{i=1}^k m^i = \frac{m^{k+1} - m}{m - 1}.$$

For just $m = 100$ and $k = 10$ already we get $N = \frac{100^{11} - 100}{100 - 1} \approx 10^{20}$ differ-

5 ent sequences. It is obvious that such a solution trial is unacceptable. For real data, distinctly less sequences are contained in all transactions of set D .

It is much more logical and quicker to test all the sequences which are included in each transaction and to define the support value for
10 each of them. Assuming the length of transaction d_0 to be m , this would mean that the transaction d_0 includes the number of all sequences of length i which equals the set of i combinations of the items above m . Then the total number of sequences N contained in a transaction of length m is:

15
$$N = \sum_{i=1}^m \frac{m!}{i!(m-i)!} = 2^m - 1$$

A transaction of length 10 encompasses 1023 sequences, a transaction of length 20 encompasses 1048575 sequences of lengths from 1 to 10 and, accordingly, 20. In spite of the fact that the numbers mentioned describe the most unfavorable case, the number of possible
20 sequences contained in a transaction may become very large.

Each of those sequences must be tested for its minimum support condition, in other words the sum of transactions including them must be formed. It is evident that this process takes a lot of time, too. For this reason, the methods according to the invention described
25 here make use of a solution trial for which no superfluous tests are needed.

Two variants of extracting large sequences will be described below. Both utilize a common record of basic concepts, but they differ in the summing of the support values of the sequences.

5 The methods belong to the category of iterative procedures. In each step of the procedures all large sequences of a given length are defined. Operation of the procedure is continued until all large sequences of maximum length have been found, in other words until no further large sequences are discovered in the current operating step. Another criterion for terminating the procedure may be the
10 reaching of a maximum length of the sequences which were predetermined by the user.

With both variants of the method according to the invention the large sequences of length k are used for constructing the set of new potential large sequences of length $k+1$. A sequence is called a candidate if it was generated from large sequences of a length reduced by 1 and if, possibly, it likewise is a large sequence. In this manner the set of candidates having length k presents the set C_k . Obviously, the following is true: $C_k \supseteq L_k$.

1. Method of generating the candidate set.

20 In both methods, the same principle of generating candidates is applied. The principle differs slightly from the one of generating candidates in the preceding IBM solution trial. The function for generating the candidates uses all large k -sequences - L_k as the input set. The result it offers is a super set of the sets of all
25 large $(k+1)$ sequences - C_{k+1} . The method of generating the candidate set may be written as stated below, making use of an SQL-like syntax:

```
INSERT INTO Ck
SELECT p.item1,p.item2,...,p.itemk,q.itemk
30 FROM p,q
WHERE p.item2 = q.item1, p.item3 = q.item2,... p.itemk = q.itemk-1.
```

The mode of generating the candidates may be illustrated as follows.

Assuming the large sequences of length 4 listed below were found in a step of the procedure:

1. 1 3 2 6
2. 1 4 1 2
- 5 3. 2 1 3 2
4. 2 2 2 2
5. 3 1 2 1
6. 3 2 6 2
7. 4 1 2 1.

10 Checking all possible pairs of large 4-sequences, beginning with the first one and including also all those consisting of the same sequence, the following candidate set is obtained:

- 1 3 2 6 2 - result of the union of 1 and 6
- 1 4 1 2 1 - result of the union of 2 and 7
- 15 2 1 3 2 6 - result of the union of 3 and 1
- 2 2 2 2 2 - result of the union of 4 and 4

Sequences 5., 6., and 7. did not provide a single pair in which they would have been the first of the "parents". Sequence 5. did not take part in the generation of candidates at all. Sequence 4. formed a
20 union with itself.

The result of the procedure of this method may bring forth a situation where not a single candidate sequence is generated by this method. This means that it is impossible to find but a single sequence of the next level, the $k+1$ level. Consequently, the method
25 ends with this step.

In the example described above, all possible pairs of large sequences of the previous level were drawn upon for constructing the candidate set. When the method works on real data, there is great likelihood of finding, in a particular step, a set of all large sequences which may consist of thousands or tenthousands of sequences.
30 The process of completely reviewing all pairs possesses a square complexity. On real data, therefore, millions or hundreds of millions of operations would have to be carried out to compare se-

quences. The process of generating candidates thus might involve some serious time expenditure.

For this reason, a data structure was elaborated which permits to reduce the complexity of reviewing the pairs of large sequences. According to this concept, the first and the last large $(k+1)$ sequence must be stored for each large k -sequence, beginning with the k -sequence under review. Of course, that requires sorting of the record of k -sequences from the small k -sequences to the large sequences. The k -sequence A is defined as being smaller than the k -sequence B if the first different item is smaller in sequence A than in sequence B .

Thus, we possess in a step of the method L_k - the set of the large k -sequences of the preceding step, C_{k+1} - the set of the candidates. For each k -sequence from L_k , moreover, the number of the second "parent" which generated this sequence is memorized. Based on this number, the spectrum of sequences from L_k with which each k -sequence may be united can be defined for each k -sequence. This data structure is employed in both variants of the method specified. A dynamic array having the dimension $|L_{k-1}| \times 2$ is used for memorizing the numbers required for the sequences from L_{k-1} .

In the k^{th} step, the set of large sequences L_k is available with each of which the number of the sequence from L_{k-1} is connected that forms the second "parent". As a result of the work the set L_{k+1} is obtained from C_{k+1} in the k^{th} step, and for the sequences of this set the second parent's numbers are memorized, and a new structure is formed of the numbers of the sequences from L_k .

In the example described above, the sequence $\langle 2 \ 1 \ 3 \ 2 \rangle$ was formed in the preceding step from the sequences $\langle 2 \ 1 \ 3 \rangle$ and $\langle 1 \ 3 \ 2 \rangle$. Assuming sequence $\langle 1 \ 3 \ 2 \rangle$ had had its place at number 3. of the list of sequences L_3 , then the numbers 1 and 1 must be memorized in the structure described for the sequence of number 3. - in accordance with the first and last sequences of L_4 which begin by $\langle 1 \ 3 \ 2 \rangle$. If the new sequence $\langle 2 \ 1 \ 3 \ 2 \ 6 \rangle$ proved to be large the number of its second

parent is memorized - 1, for sequence <2 1 3 2> it is number 3 - the number of the first newly obtained 5-sequence.

If a descendant does not exist for a sequence from L_{k-1} this fact is memorized in the structure of the numbers by way of an invalid number - for example 1.

2. The first variant

The process of operation of the first method may be subdivided into a plurality of stages:

- (i) search for all large 1-sequences - the various items of $E - L_1$.
- 10 (ii) search for all large 2-sequences - L_2 on the basis of L_1 . For each large sequence found, the list of transaction numbers containing the same is memorized at the same time.
- (iii) search for the large k -sequences on the basis of L_{k-1} . The third stage is repeated until $L_k = \emptyset$ has been obtained.

15 The first variant for which a flowchart is shown diagrammatically in fig. 3 may be represented as follows in pseudo code:

```
L1 = {large 1-sequences};
C2 = candidate-gen(L1);           // 2 candidates
forall transactions t ∈ D do begin
20   Ct = subset(C2,t);           // candidates contained in t
   forall candidates c ∈ Ct do
       c.count++;
   end
L2 = {c ∈ C2 | c.count ≥ minsupp};
25 for (k=3; Lk-1 ≠ ∅; k++) do begin
   Ck = candidate-gen(Lk-1);
   forall candidates c ∈ Ck do begin
       Tc = subset(c,D);         // transactions including c
       c.support = |Tc|;
30   end
   Lk = {c ∈ Ck | c.count ≥ minsupp};
end
answer = ULk;
```

In the first variant of the method, two solution trials are combined
35 for counting the support values of the candidate sequences. Sequential transit through the set of input transactions is the method employed to find set L_2 . For each transaction, the candidates con-

tained in it are defined, and for each candidate the counter of the support value is incremented by one.

In the first variant of the method the same solution trial was selected for this stage and for the next one: for each candidate, the
5 list of transactions is determined which might contain the candidate. The list of transactions results as the cut set across the memorized transaction lists of the parents of the candidate under investigation. Evidently, the candidate sequence can be included
10 only in those transactions in which both parents of the particular candidate are memorized. The list of transactions of the candidate includes the numbers of the transactions and, making use of the cut set formation, only those numbers are added which are included in the lists of transactions of both its parents. The list thus obtained permits the number of transactions which must be tested for
15 the presence of a certain candidate to be reduced substantially, especially so in the final steps of the procedure.

However, transit through the whole set of candidates is a precondition for the statement of the transaction lists. In the second stage of the method, when counting set L2, nothing but the items of set L1
20 are known. Any pair of large 1-sequences forms a candidate of length 2. During operation of the method with real data, the power of set L1 may reach several tens of thousands of 1-sequences. The number of candidates of length 2 then will be a few hundred million. A complete review of all the possible candidates would require an unreasonable amount of time. In this case it makes sense to select the
25 principle which was used fully in the second variant of the method.

According to the basic idea of the second variant of the method, a counter of the support value is introduced for each candidate. That is followed by the sequential transit through the entry set of the
30 transactions. For each transaction, the candidates it includes are defined, and their counters of the supports are incremented by one. In the second variant of the method, a more complex scheme is used for defining the candidates which are contained in a particular transaction, but a simpler method may be resorted to for defining
35 set L2. Two transits through the transactions, one embedded in the

other, are required for defining the numbers of all candidates. In the outer cycle, the large items contained in the transaction are selected one after the other. The inner cycle begins with the item which succeeds the item chosen in the outer cycle. Each selected
5 pair of large items of set E represents a length-2 candidate and, therefore, the corresponding counter of the support value of the candidate must be incremented. The number of the counter is calculated starting from the indices of the selected items in the list of large items and the total number of large items found. Let a and b -
10 the respective selected first and second items of a candidate and N - be the number of 1-sequences found in the preceding stage of the method - the large items from L1. Then the number of the respective candidate is obtained according to the following formula: $n_c = Na + b$

If the repeated incrementing of the counter of the support of a specific candidate is to be excluded, in dealing with one and the same
15 transaction, a means of marking the fact that incrementation of the counter took place in a particular transaction must be provided. The simplest way of doing that is to store, for each candidate, not only its counter of the support but, in addition, also the number of the
20 last transaction which caused the counter of the candidate of interest to be incremented. In view of the fact that the transactions from the entry set are dealt with successively, the content of the field of the number of the last transaction gives an indication whether the counter was incremented since the beginning of dealing
25 with the concrete transaction under review.

The manner of proceeding as described above for determining the 2-sequences L2 has the advantage that the calculations involved are relatively simple because only those candidates are processed which indeed are contained in the transactions input. Yet this method also
30 suffers from a serious disadvantage. For each candidate, two values related to it must be stored: the counter of the support value and the number of the last transaction which incremented the counter. Using a 32 bit word for each field which, theoretically, would allow processing a maximum of 2147483647 transactions, requires 8 bytes
35 per candidate. At the stage of defining set L2 there may be hundreds of millions of different candidates. This stage in the operation of

the method consequently requires some hundred million bytes of RAMs and that, indeed, presents a serious limitation. It is the square-law dependence of the memory needed on the number of large 1-sequences found in the preceding stage that causes this serious
5 limitation.

To overcome this limitation, a scheme of multiple transit of the processing of the transaction set in the course of constructing set L2 was realized in the final version of the first method. During a full transit through the input set of transactions, only those pairs
10 are dealt with whose first item lies within a certain range of values. In this manner the storage space required for the second stage of the procedure can be restricted, in consideration of the storing capacity available in the computer. Of course, greater memory restrictions in turn will demand more transits and, accordingly, the
15 procedure will become slower in the second stage. The method described of defining the 2-sequences L2, therefore, is a compromise between the necessary storage space and the processing speed. This fact is confirmed by the experimental execution time behavior of the method.

20 2.1 Further implementation details of the first variant

In the first stage of operation of the method all items of set E must be found that occur in the input transactions no less than a predetermined number of times. As the numbers and frequencies of the various items from E are not known initially a method must be pro-
25 vided of storing both the actual item and the number of transactions in which it is contained. When practicing both variants of the method, a transit through all the input transactions takes place. A standard class (e.g. from Java) - the hash table, is drawn upon for storing the items and their counters. This class permits storing of
30 the pair <key, value>, and for quick search of the values it uses the mechanism of the hash table in respect of the key values. In implementing the method, the key chosen is an object of the integer class (wrapper class for integral values) (e.g. from Java) which is initiated with the value of the selected item, and the value chosen
35 is another object of the integer class which is initiated with the value of its counter. When selecting another item of a transaction,

access is taken to the object of the hash table class to obtain the value through the key which is identical with the selected item. If such a value was found in the hash table, another rewrite, incremented by one, is caused. If no such value was found for the key, the pair to be registered in the hash table will be the pair of which the value it contains is the initial value of the counter, in consideration of the item already selected - 1.

Repeated incrementing of the counter of a transaction is avoided, in implementing the method, by choosing a different class (e.g. from Java) - vector, in which all those items are stored that were already processed by a transaction. Before searching for the value in the hash table, the method attempts to find the item in the instance of the vector class. If such an item was found, it means that it was processed before in this transaction, and the method will proceed to the next item of the transaction. Of course, the vector of values is purged when the method proceeds to the next transaction. Having gone through the transaction set, only those items are selected whose counter values are not smaller than the minimum value. The large items found are stored in the array and sorted so that, in the further course of proceedings, binary search can be applied to the items and be mapped to the set of integral positive numbers 0, ..., N-1, with N being the number of large items found. The large items are used to form set L1 of the large 1-sequences.

In implementing the method, the *TransactionList* class is used for storing the list of the numbers of the transactions. It permits storing the numbers of the transactions in an array, adding a number to the end of the list, provided the numbers are added in rising order. This is an indispensable condition for the correct operation of the method of the *TransactionList* class which forms the cut set of both transaction lists. The initial lists of the transactions are formed in the course of the sequential transit through the set of input transaction during the stage of the construction of L2, at the same time, establishing, in a natural way, the conditions of the sorting order. Further, the new lists are created while the cut set operation is executed. The lists of the numbers are stored in an array. However, since the length of the array is defined during the

stage of dynamic storage allocation, additional fields of the array must be made available, as required, by increasing the length. An array having an initial size is made up and gradually filled during the generation of an instance of the *TransactionList* class. When the maximum of numbers is reached, the memory is made available once more. The maximum storage capacity of the array is increased ten-fold, for instance, a new array having the same maximum length is made available and the contents of the previous array are copied into the same. The old array may be marked as superfluous (e.g. for the virtual machine in Java).

During the third stage of operation of the method, all the large k-sequences memorized during the preceding step are reviewed. The numbers of the first and last large k-sequences, with which the sequence found can be united, are found by way of the stored number of the second parent of the sequence, making use of the data structure described above. Moreover, the intersection forming operation is carried out across the lists of transactions of both sequences. If the length of the resulting list is smaller than the required minimum support value the candidate under review is discarded immediately even without an operation of a direct candidate and counting of the number of transactions containing the same. Otherwise, the sequence of the items of the candidate itself is generated and each transaction of the list obtained from the formation of the intersection is checked as to the content of candidate sequences it includes. If a corresponding sequence was found in a transaction the number of the transaction is memorized. In this manner, at the end of processing a candidate, there will be a list of transactions which include the candidate. This candidate will be stored together with the number of its second parent if the number of such transactions is not smaller than the minimum number.

The actions described above are repeated for each candidate. If, ultimately, not a single large sequence was found the operation of the third stage of the method is terminated and the results are output via the interface. There may be two reasons why not a single sequence was found in the course of one step. In set *L_k*, there may not be a sequence which might form a candidate together with another se-

quence, i.e. not a single candidate exists in a step. Another reason may be the variant where, ultimately, not a single candidate presents a large sequence.

Summarizing, the first variant of the method proves to be quick on large amounts of data, and its storage space requirement is small. As far as storage space and computing time regarding the use of transaction lists are concerned, empirical calculations and experimental results show that, based on real data, they tend to be greatly reduced because, as the sequences being processed become longer they will be contained in a progressively smaller number of transactions, as a result of which not only the lists become shorter but also the amount of time needed to process them. Consequently, it is a requirement for operation of the first variant of the method that the initial set of transactions be entered in the form of a list of sequences of integral numbers, the minimum support value be input as a floating value, and the storage space available for the second stage of the method be input, too.

3. Second variant

The second variant of the method, of which a flowchart is illustrated diagrammatically in fig. 4, was elaborated and implemented for purposes of comparison with the first variant of the method as regards speed and storage space. The second variant can be represented as follows in pseudo code:

```
L1 = {large 1-sequences};
25 for (k=2; Lk-2 ≠ ∅; k++) do begin
    Ck = candidate-gen(Lk-1); // new candidates
    forall transactions t ∈ D do begin
        Ct = subset(Ck, t);
        forall candidates c ∈ Ct do
30             c.count++;
    end
    Lk = {c ∈ Ck | c.count ≥ minsup}
end
answer = U Lk;
```

35 The method illustrated is very general and does not implement the process of selecting the candidates that are included in a particular transaction. In the simplest case this may be reviewing all can-

didates with the aim of selecting those which are contained in the sequence of the transaction. Another variant provides for reviewing all sequences of a given length and selecting those from among them which represent candidates. It is evident that both methods are disadvantageous for being inefficient. Therefore, a different method is described for implementing the second variant.

The second variant makes use of quite a few of the same ideas as the first one. In particular, the same principle is applied without much alteration for calculating set L1. The entire description given above of the first stage of operation of the first variant of the method applies to the second variant as well. As already mentioned, for adding the support of the candidates, the second variant of the method makes use of the concept which, in the first variant, was applied only in the second stage for generating the set of large 2-sequences, L2.

3.1 Further implementation details of the second variant

A specific data structure is used with the second variant of the method so as to have the possibility of defining the list of candidates included in it, for every transaction. From the second stage of the method on, each transaction is presented as a list of the numbers of the k-sequences found in the preceding step of the method. Reviewing the pairs of these k-sequences and uniting them according to the method described above of generating candidates, all the candidates of length k+1 can be obtained that are included in a particular transaction. Yet the list of the large k-sequences is organized as a two-dimensional array of values so as to separate the candidates. The resulting data structure will be illustrated in the example below.

Let us assume the entry set of transactions includes the following transaction:

<3 1 3 2 1>.

Furthermore, let it be assumed that in the second stage of operation of the method it was discovered that the sequences <3 1>, <3 2>, <1 2>, and <1 1> contained in that transaction are large, having num-

bers 5, 7, 3, and 1, respectively. In the second stage of operation of the method the following data structure was generated, which corresponds to the given transaction:

1. 5 7
5 2. 3 1
3. 7 5
4. -

The number of lines of this structure equals the number of items in the transaction during the stage of construction of set L2. In the further stages this set equals the number of lines in the preceding structure for this transaction. Each line contains the numbers of the large sequences which begin with this item or with this line in the structure that was generated in the preceding step. In each step, the pairs of numbers of the large sequences of the previous level are reviewed, and the method searches all pairs of numbers which might be united. The test for possible union is carried out on the basis of the same principle as with the first variant of the method.

A structure of this kind is formed for each transaction and is updated in each step of the method. The first structure is generated in the second stage of the method, and that is the only difference compared to the corresponding stage of the first variant of the method. In one step of the method, all transactions of the entry set are processed successively, utilizing the structure just described. Step by step each item of each line is selected, and following that the items of each subsequent line are searched. If the pair thus selected may become a candidate the counter of its support is incremented by one. To avoid multiple incrementing of the counter during the processing of a transaction, the idea of the first variant of the method may be resorted to in that the number of the last transaction incremented is stored for each candidate.

This illustrates that actually it is not necessary, with the second variant of the method, to generate candidate sequences during the execution time because, contrary to the first method, that is not needed for adding up the support values. The large k-sequences found

during the k^{th} step are stored in the form of the pairs of their parents. That helps save storage space because in practice the large k -sequences may come about at the end of the operation proper of the method during the transit from set L1 to set L2 so that they will
5 not unnecessarily occupy storage space while the main cycle of the method is underway.

The process of filling the required structure may turn out to be difficult because, while the numbers of the candidates are known during the processing of the transaction, those of the large sequences are not as they are defined only after all transactions have
10 been processed. During the processing, therefore, the numbers of the candidates contained in them are entered into the renewed structure. When all transactions of the entry set have been processed the numbers included in the structure must be corrected to allow for the
15 candidates which did not fulfill the support condition. The part of the method during which the numbers are corrected may be presented as follows:

```
m = 0;      // accumulator of the correction
forall c ∈ Ck do begin      // for each candidate
20   if c.count ≥ minsup then c.correction = -m-1
      else m++;
end
```

Having accomplished this fragment of the method, the value of the correction is known for each candidate and must be deducted from the
25 number of the candidate so as to provide the number of the large sequence. In fact, to get the number of the sequence from the number of the candidate the number of candidates that did not make the test must be deducted whose numbers are smaller than the number of the candidate in question. The final stage of the renewal of the struc-
30 ture of the transactions is the transformation of the numbers of the candidates into the numbers of the large sequences simultaneously with the removal of the numbers of the non-large candidates and the sorting.

The sorting serves to expedite the search for paired sequences during the processing of the transaction structures, in other words
35 those sequences with which the selected sequence may be united.

Without sorting of the lines of the structure of the transactions the search for the second parent would involve a full transit of the subsequent lines, including a test of the condition for union. Sorting the values of the numbers of the large sequences introduces a
5 rising order, thus providing the opportunity of most rapidly finding a value in the line, by binary search, starting with which the numbers of the parent sequences are memorized. Moreover, an orderly transit through the line is carried on until a number of a sequence is larger than the maximum parent number for the selected first parent.
10 Experimental data have demonstrated that the sorting introduced, which is not absolutely required for the operation of the method, accelerates operations in those cases in which the structure related to a particular transaction contains some very long successive lines. With the first variant, without sorting, a clear drop in
15 efficiency could be observed in some parts of the transaction set, which is why the overall time of the operation of the method is much prolonged. Having applied the sorting, the method operated many times faster on such data, depending on the particular entry transaction sets.

20 During the processing of the transaction structures, the numbers of the large sequences found in the preceding step of the method and included in the transaction currently being processed are selected successively. For each number selected it becomes a task to define the numbers of the first and last parent with which the sequence
25 having the selected number may be united. Besides, the number must be known for every candidate in order to be able to increment the counter. This problem is solved partly as was done with the first variant of the method. Yet the implementation of the second variant makes use of a slightly modified idea in consideration of the requirement that the number of the candidate must be known. In the
30 case of the first variant, the number of the candidate is known because, for all practical purposes, in the main cycle a transit through the candidate set takes place and a specific structure is renewed in each step for quickly determining the first and last numbers of possible parents. With the second variant, a modified structure is applied in consideration of the need for storing the numbers
35 of the corresponding candidates. In each entry of this structure,

not only the numbers of possible parents are stored but also the number of the first candidate that can be generated. Via the number of the second parent of a large sequence, thus the numbers of the parents can be determined with whom the sequence under review may be united, and also the numbers of the corresponding candidates.

With the second variant of the method as, by the way, also with the second stage of the first method, storage space must be allocated to the values of the counter of the support and the array of the numbers of the transaction which was the last to increment the counter.

10 4. Description of a program interface

The program interface is considered in the form of an example of applying the method in a Java program.

The *TransactionSetSeq* class is used for storing the record of the sequential entry transformations. An instance of the class is realized by the following instruction:

```
TransactionSetSeq tss = new TransactionSetSeq();
```

Now we add all transactions of the entry record:

```
for (int i=0;i< <number of transactions>;i++) {
```

Each sequential transaction is represented as an instance of the *ItemSetSeq* class:

```
ItemSetSeq iss = new ItemSetSeq();
```

The items of the transactions are integral numbers - the following method of the *ItemSetSeq* class serves to add a new item to the transaction:

```
25        void addItem(int item);
```

We add all the items of the current transaction:

```
for (int j=0;j< < number of items of the current transaction>;  
j++)
```

```
iss.addItem(<number of the item >);
```

The following method of the *TransactionSetSeq* class is used for adding a transaction to the transaction record:

```
void addTransaction(ItemSetSeq iss);  
tss.addTransaction(iss);  
5 }
```

An instance of the *Sequential* or *Sequential2* class must be generated for calculating the sequences:

```
Sequential seq = new Sequential();  
Sequential2 seq2 = new Sequential2();
```

- 10 The following method contained in the *Sequential* and *Sequential2* classes must be called to start the method:

```
SequentialResult SequentialAlg(TransactionSetSeq tss, double  
minsupp);
```

where

- 15 *tss* - entry set of the transactions,
minsupp - minimum support value.

```
SequentialResult seqResult = seq.SequentialAlg(tss,minsupp);
```

or

```
SequentialResult seqResult = seq2.SequentialAlg(tss,minsupp);
```

- 20 The result of operation of the method is stored in the object of the *SequentialResult* class. The *SequentialResult* class contains the following method for accessing the sequences found:

```
ItemSetSeqList getLargeSequences();  
ItemSetSeqList issList = seqResult.getLargeSequences();
```

- 25 The *ItemSetSeqList* class contains the methods:

```
int getSize(); - supplying the number of sequences,  
ItemSetSeq getItemSetAt(int index); - supplying the sequence of  
the number index.  
for (int i=0; i < issList.getSize(); i++) {
```

```
ItemSetSeq iss = issList.getItemSetAt(i);
```

The ItemSetSeq class contains the following method for accessing the sequence:

```
int getItemAt(int index);
```

5 The following method supplies the length of the sequences:

```
int getSize();
```

```
for (int j=0;j<iss.getSize();j++)
```

```
int item = iss.getItemAt(j);
```

```
}
```

10 The following method of the ItemSetSeq class supplies the value of the counter for the support:

```
int getSupportCount();
```

15 The actual support value support for each sequence is the quotient of the value of the support counter and of the number of all transactions.

5. Example: Clickstream analysis

Referring to figs. 1 and 2, the application of the method described above will now be explained by way of example with reference to a clickstream analysis of Web servers.

20 The user needs a Web browser 10 to call a Web site via the internet, for example the Netscape Navigator, by which he connects to a Web server 20, for example the Apache Web server. Information about HTML pages of a Web site that were fetched is stored by Web servers in data bases 30 or log files 40. The information concerning those HTML
25 pages is stored in a table. The columns correspond to the information about the page that was fetched, e.g. date and time, URL of the HTML page fetched, referer URL, type of Web browser. The lines correspond to the fetched HTML pages themselves. Various mechanisms, not described in any greater detail here, make it possible to identify
30 Web sessions and allocate them to each fetched page. A Web ses-

sion is a sequence of HTML pages which a user has fetched successively from the Web site.

It is the aim of the clickstream analysis to find out typical paths along which users of a Web site mainly move. For instance, it may happen with an E-commerce shop that many users enter the shop through the main page, go into a certain product category, look at a certain product in a detail view, and thereafter leave the shop. The conclusion may be drawn that the product looked at did not satisfy the demand, and that consequently its presentation absolutely needs to be improved. Where users choose cyclical paths, often the conclusion may be drawn that for them the shop is not structured clearly, and so on. Now, having provided a data access module 50, a data mining application 60 implemented via a program and performing the actual clickstream analysis takes access to the data of the HTML pages fetched and to be found either in the data base 30 or in the log files 40. The data access module 50 executes various transformations, like the resolution of IP addresses, recognition of date formats, and the extraction described above of Web sessions.

The mining application 60 calculates the most frequent paths followed by users through the Web site. Finding the typical paths in Web sites is a task for sequential analysis. In this case $E = \{e_1, e_2, \dots, e_m\}$ is the set of all HTML pages of a Web site. Each transaction d_i corresponds to a Web session, the items of which form a sequence $\langle s_0, s_1, \dots, s_k \rangle$ of HTML pages. The total number of transactions D corresponds to the total of Web sessions of the Web server. The clickstream transactions for a simple example are shown at 80 (after mappings 90 and 100 into the range of integers) and 110, respectively. It is the task of the clickstream analysis to find all the large sequences of HTML pages which provide the mining results 70, with a given minimum support parameter s . In the example of the clickstream table 80, the large sequences found are shown at 120 or 130, respectively, (after re-mapping) for a minimum support value of 50 %. These are the typical users paths of the Web site whose analysis, as explained above, may be used for improving the structure of the Web site. In particular, the results may be incorporated di-

rectly in the Web server, for instance, by way of recommended HTML pages (recommender systems).

5 The method described of the invention is very well suited for calculating large sequences, especially so because it can analyze large amounts of data. Log file tables of popular Web sites, in particular, often comprise millions of HTML pages fetched and, therefore, are extremely demanding as regards sequential analysis methods. Consequently, evaluating this information by means of sequential analysis usually was very intensive in terms of computer time and thus
10 limited to special cases (sequences of length 2, only a few Web sessions, etc.). These restrictions are overcome by the method described here for almost any clickstream data occurring on present day Web servers.

15 The features of the invention disclosed in the specification above, in the claims and drawing may be essential to implementing the invention in its various embodiments, both individually and in any combination.